Assignment 2

STAT 4620/5620 Winter 2025

Submission Instructions

- Follow the instructions in Question 1 to create an R package with version control tracked by Git and upload it to GitHub. Include a link to the GitHub repository at the end of Question 1.
- Modify this Quarto document to answer the questions and upload your Quarto file and rendered PDF file to Brightspace.

Question 1 (14 points)

- This question will guide you through the creation of an R package that can be used to increase the reproducibility and shareability of your data analyses.

1. Install the `devtools` package.
2. In R Studio click the "Create a Project → New Directory → R Package using devtools". Use the directory name "A2NAME", replacing NAME with your name. Feel free to pick any folder for "Create project as a subdirectory of:", a good option is your Data Analysis STAT4620/5620 folder for the course, if you have one. Click "Create Project". R Studio should automatically switch to the newly created project, switch to it yourself if not.
3. Edit the package DESCRIPTION file to write your name in the author field.
4. Start using version control for your package.

    1. Install git if you have not already. Unless you know what you're doing, when the git installer asks you to choose the default editor used by git, switch it from Vim to something more familiar like Notepad. Use the defaults for the rest of the options. Restart Rstudio.
    2. Find the "Terminal" pane in R Studio and type in the following commands, substituting in your own name and e-mail:
        - `git config --global user.name 'Ethan Lawler'`

    - `git config --global user.email 'lawlerem@dal.ca'`
    3. In R Studio press "Tools → Version Control → Project Setup…". A pop-up window should show up with the Git/SVN menu selected. Change "Version control system…" to "Git", and say "Yes" when it asks you if you want to initialize a new git repository for this project, restarting R Studio if necessary.
    4. Find the "Git" panel in R Studio. This panel lists the files and folder in your package and gives you some information about each of them. Check the box for each file under the "Staged" column and make sure the "Status" column changes to a green A that says "Added" if you hover your cursor over it.
    5. When all files are added press the "Commit" button. In the pop-up window write "Initial commit" in the "Commit message" box. Press the "Commit" button, then feel free to close all the windows except for your main R Studio window.
    6. Still in the "Git" panel, press the "History" button. You should see a pop-up window containing a line saying "Initial commit" along with your name, e-mail address, and date that you completed the previous step. Close the pop-up window.

5. In the R folder of your package create an R file named `mean.R`. The `mean.R` file should contain a function `mymean` that takes a numeric vector as input and returns the mean of the vector. Make sure your `mymean` function is fully documented using tags, including a title, the name and description of the input vector, and a description of what the function returns. Make sure your documentation includes a tag to export your function. Run `devtools::document()` in the "Console" panel to create help pages from your documentation. You should notice that NAMESPACE, R/, and man/ now appear in the "Git" panel. Add them by checking the boxed under the "Staged" column, the NAMESPACE status should change to a blue M to show that the NAMESPACE file was already part of the git repository but has been changed since the previous commit. Commit the files and folders using the commit message "Added and documented a function to compute a mean."
6. Load the *cars* data.frame included in R then run `usethis::use_data(cars)` to add the *cars* dataset to your package. You should notice that a "data" folder has been added to your package. Add a *data.R* file to the R folder of your package and document the cars dataset there. Run `devtools::document()` to create help pages from your documentation. Add all of your new files and create a commit using the message "Added and documented a copy of the cars dataset."
7. Create a package vignette with the code `usethis::use_vignette("A2NAME.qmd")`, replacing NAME with your name. You can use a package vignette to write a reproducible data analysis that uses the function you write in your R/ folder and the data you add to your package. For now you can leave the vignette empty. Add all the files to git and commit them with the message "Completed Question 1!".

Now we'll set up your GitHub account and post your R package on GitHub. This step will be a bit complicated but you only ever need to do this set-up once. We will - Create a github account and an automatic password called a personal access token. - Create an ssh key that lets you move things from your computer to GitHub. - Create a new repository on GitHub then upload your package to GitHub.

8. Go to github.com and create a GitHub account (or use an existing one). To upload to GitHub you need a Personal Access Token, go to this link and follow the instructions for "Creating a personal access token (classic)". When you get to step 8 and are asked to select the scopes you'd like to grant, make sure the "repo" box is checked. Click the clipboard icon next to your newly created personal access token to copy it. In Rstudio, install the "gitcreds" package the run `gitcreds::gitcreds_set()`. When it asks you to "enter password or token:" paste in the copied personal access token and press enter. Restart RStudio.
9. Back on GitHub, go the the "Your repositories" page and click the "New" button to create a repository for your R package. The repository name should have the same "A2NAME" name as your package. Keep the default option to make it a public repository and click the "Create repository" button.
10. In the "Quick setup" box, make sure the "HTTPS" button is selected. Copy the first two lines of the box under "…or push an existing repository from the command line". The first line should contain a url for you new repository that starts with "https://". Open the Terminal panel in Rstudio, paste in those contents and press enter. In the Git panel of Rstudio, press the "Push" button to upload your files to your newly created GitHub repository.
11. Go to your package repository on Github and check to see that all of the files in your package are now uploaded to GitHub. Copy the url for your GitHub repository and paste it below.

GitHub Repository URL:

Question 2 (4 points)

- Explain how the Akaike information criterion (AIC) is computed for a generalized linear model and how it is commonly used for model or variable selection purposed. Be sure to describe the two competing goals that AIC tries to find a good balance for. (250 words)

Question 3 (4 points)

- Residual checking for GLMs is not always as straightforward as it is for linear models, and the problems are particularly acute in the case of binary responses. Explain why (100 words)

Question 4 (16 points)

- We are interested in a study concerning lung function in patients with cystic fibrosis (Altman 1991, p.338). Install the **ISwR** package and load the *cystfibr* dataset provided in that package.

  1. Fit a model relating maximum expiratory pressure (pemax) to the explanatory variables contained in the dataset. Describe the steps you take including fitting an initial model, residual analysis, and changes to your initial model if needed. Make sure you include the reasons why you make certain modelling choices.
  2. Interpret results for the sex variable.
  3. Try using the *step* function and interpret the results.
  4. What can you reasonably conclude from your analysis?

```r
if( !requireNamespace("ISwR", quietly = TRUE) ) install.packages("ISwR")
library(ISwR)
data(cystfibr)
```

Question 5 (10 points)

Show how to parameterize the following piecewise linear spline to ensure that the resulting function is continuous:

1. Use the knots $c = 10, 30, 50, 75$ so that you will have three linear pieces.
2. Let the slope of each linear piece be free to vary then find conditions for the intercept of each linear piece to ensure the function is continuous.

- *Hint:* It might be easier to write each linear segment in the form $f_i(x) = \beta_{0,i} + \beta_{1,i}(x - l_i)$ where l_i is the left endpoint for that linear segment.

3. Modify the function below so that it takes a vector beta and an input x and evaluates the spline using your parameterization. You might need to remove β_5 and β_6 and replace them with a value using your answer to (2.)
4. Write a sum-of-squares function that takes a β parameter vector as input and gives you the sum-of-squared errors as output, using cystfibr as your dataset, pemax as your response variable, and weight as your covariate. Then use the nlminb function to find the best estimate for β and plot the estimated piecewise linear function in the same plot as the data.

```r
knots<- c(10, 30, 50, 75)
f<- function(beta, x) {
    which_piece<- x |> cut(knots, labels = FALSE)
    linear_pieces<- list()
    linear_pieces[[1]]<- function(x) beta[1] + beta[2] * (x - knots[1])
    linear_pieces[[2]]<- function(x) beta[5] + beta[3] * (x - knots[2])
    linear_pieces[[3]]<- function(x) beta[6] + beta[4] * (x - knots[3])
    prediction<- sapply(
        x |> seq_along(),
        function(i) linear_pieces[[which_piece[i]]](x[i])
    )
    return( prediction )
}
```